PATENT ABSTRACTS OF JAPAN

(11)Publication number : 04-270431
(43)Date of publication of application : 25.09.1992

(51)Int.Cl. G06F 12/08
G06F 12/08

(21)Application number : 03-085843
(22)Date of filing : 27.03.1991

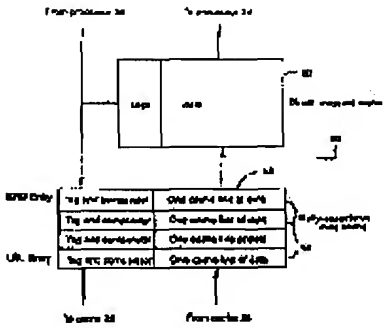
(71)Applicant : DIGITAL EQUIP CORP <DEC>
(72)Inventor : JOUPPI NORMAN P
EUSTACE ALAN

(30)Priority

Priority number	Priority date	Priority country
90 499958	27.03.1990	US
90 500062	27.03.1990	US

(54) MEMORY SYSTEM FOR DATA PROCESSOR

(57)Abstract:
PURPOSE: To reduce the miss penalty by improving the performance of a cache memory.
CONSTITUTION: A small complete extension miss cache 42 is provided between a cache 20 of a first level and a cache 26 of a second level to utilize miss caching. Though a miss penalty in many cycles is given without the miss cache 42, a miss in the cache 20 hit in the miss cache 42 has a miss penalty only in one cycle. A improved method of miss caching where the small complete extension miss cache is victim caching. A stream buffer is started with a cache miss address to prefetch a cache line. It is useful to remove a cache miss of a large capacity.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision

of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平4－270431

(43)公開日　平成４年(1992)９月25日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　12/08		Ｆ　7232－5Ｂ		
	３１０	7232－5Ｂ		

審査請求　有　　請求項の数33(全 19 頁)

(21)出願番号　特願平3－85843

(22)出願日　平成３年(1991)３月27日

(31)優先権主張番号　０７／４９９９５８

(32)優先日　1990年３月27日

(33)優先権主張国　米国（ＵＳ）

(31)優先権主張番号　０７／５００００２

(32)優先日　1990年３月27日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　590002873
デイジタル　イクイプメント　コーポレイション
アメリカ合衆国　マサチユーセツツ州　01754メイナード　メイン　ストリート　146

(72)発明者　ノーマン　ピー　ジヨウピイ
アメリカ合衆国　カリフオルニア州　94306　パロ　アルト　コロラド　アベニユー　617

(74)代理人　弁理士　杉村　暁秀　(外５名)

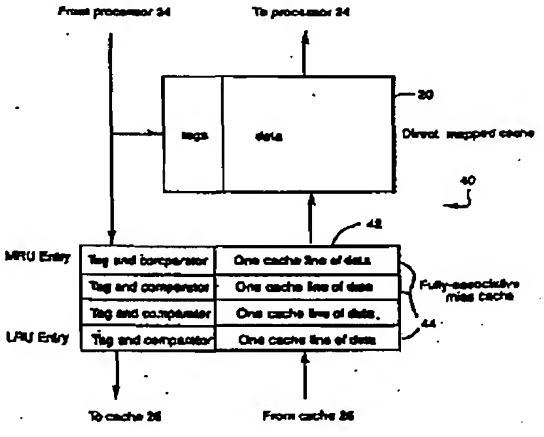
最終頁に続く

(54)【発明の名称】　データ　処理装置のメモリ　システム

(57)【要約】　(修正有)

【目的】キャッシュメモリのパフォーマンスを向上して、ミスペナルティを減少させる。

【構成】第１レベルのキャッシュ２０と第２レベルのキャッシュ２６との間に小さい完全付属ミス　キャッシュ４２を設けてミス　キャッシュすることを利用する。ミス　キャッシュ４２がないならば多数サイクル　ミス　ペナルティを持つのに反して、ミス　キャッシュ４２中でヒットするキャッシュ２０内のミスは、僅かに１サイクル　ミス　ペナルティしか持たない。小さい完全付属ミス　キャッシュを搭載したミス　キャッシングの改良型がビクティム　キャッシングである。ストリーム　バッファはキャッシュ　ミス　アドレスでスタートして、キャッシュ　ラインをプリフェッチする。これは容量の大きいキャッシュ　ミスを除去するのに有用である。



1

【特許請求の範囲】

【請求項1】第1キャッシュ　メモリ（18，20）と、第2メモリ（26）と、これらの第1キャッシュ　メモリと第2メモリの間に接続されたストリーム　バッファ（62）とを有し、さらに前記第1キャッシュ　メモリ、第2メモリ及びストリーム　バッファに接続されている手段（16）を有し、情報をアドレスする第1キャッシュメモリ内にミスが生じたときは、該手段は第1キャッシュ　メモリ及びストリーム　バッファに情報をアドレスし、かつ情報を供給し、該情報のアドレス及び供給手段は、前記第1キャッシュ　メモリ内でミスされた情報を供給し、かつこのミスされた情報に対するアドレスに後続する少くとも1つのアドレス内の情報を前記ストリーム　バッファに供給する手段を含んでなるデータ処理装置のメモリシステム。

【請求項2】前記第1キャッシュ　メモリがインストラクション　キャッシュ（18）を含む請求項1記載のメモリ　システム。

【請求項3】前記第1キャッシュ　メモリがさらにデータ　キャッシュ（20）を含む請求項1記載のメモリ　システム。

【請求項4】前記情報のアドレス及び供給手段は、前記第1キャッシュ　メモリとストリーム　バッファとに同時にアドレス情報を供給する手段を含んでなる請求項1記載のメモリ　システム。

【請求項5】前記情報のアドレス及び供給手段は、前記第1キャッシュ　メモリとストリーム　バッファとに同時に対応情報を供給する手段を含んでなる請求項1記載のメモリ　システム。

【請求項6】第1キャッシュ　メモリと第2メモリの間に接続された付加的な複数個のストリーム　バッファを有し、前記情報アドレス及び供給手段は、第1キャッシュメモリ内でミスされたデータを供給し、このミスされたデータのアドレスの後続の少くとも1つのアドレス内のデータを前記付加的の複数のストリームバッファに供給する手段を有してなる請求項1記載のメモリ　システム。

【請求項7】情報のアドレス及び供給手段が、ストリーム　バッファの1つ及び第2メモリよりミスされた情報を供給する手段を含む請求項1記載のメモリ　システム。

【請求項8】前記第1キャッシュ　メモリを直接マップ・キャッシュ　メモリとする請求項1記載のメモリ　システム。

【請求項9】第1キャッシュ　メモリ（18，20）、第2メモリ（26）、及び第1キャッシュメモリより小容量で、第1キャッシュ　メモリと第2メモリの間に接続された付属のミス　キャッシュ（42，52）とを有し、さらに前記第1キャッシュ　メモリ、第2メモリ、及び付属のミス　キャッシュに接続され、アドレスされた情報に

2

対し第1キャッシュ　メモリ内にミスが生じたときは、この第1キャッシュメモリ及び付属のミス　キャッシュに情報をアドレスし、情報を供給する手段（16）を具えてなるメモリ　システム。

【請求項10】第1キャッシュ　メモリがインストラクション　キャッシュを有する請求項9記載のメモリ　システム。

【請求項11】第1キャッシュ　メモリがさらにデータ　キャッシュを有する請求項10記載のメモリ　システム。

【請求項12】前記情報のアドレス及び供給手段は、第1キャッシュ　メモリ及び付属のミスキャッシュに同時に情報のアドレスを供給する手段を含んでなる請求項9記載のメモリ　システム。

【請求項13】前記情報のアドレス及び供給手段は、対応の情報を前記第1キャッシュ　メモリ及び付属のミス　キャッシュに供給する手段を含んでなる請求項10記載のメモリ　システム。

【請求項14】前記情報のアドレス及び供給手段は、前記付属のミス　キャッシュの1つ及び前記第2メモリより第1キャッシュ　メモリへ情報を供給する手段を含んでなる請求項10記載のメモリ　システム。

【請求項15】前記第1キャッシュ　メモリが直接マップ　キャッシュ　メモリであり、情報のアドレス及び供給手段は、前記付属のミス　キャッシュの1つより情報を移動された直接マップ第1キャッシュ　メモリ及び第2メモリより前記の付属ミスキャッシュに情報を供給する手段を含む請求項14記載のメモリシステム。

【請求項16】情報のアドレス及び供給手段が、パリティ　エラーの際、正しい情報を、前記付属ミス　キャッシュより直接マップ　第1キャッシュ　メモリに供給する手段を有する請求項15記載のメモリ　システム。

【請求項17】インストラクションの一部分を付加的に含んでいるデータを前記情報が有しており、メモリ　システムは、前記直接マップ　第1キャッシュ　メモリと第2メモリの間に接続されたストリーム　バッファを付加的に有しており、前記情報のアドレス及び供給手段は、前記直接マップ　第1キャッシュ　メモリ内でミスされたインストラクションを供給し、かつこのミスされたインストラクションのアドレスに後続する少くとも1つのアドレスにおける情報を、該ストリーム　バッファに供給する手段を有してなる請求項15記載のメモリ　システム。

【請求項18】メモリ　システムがさらに、前記直接マップ　第1キャッシュ　メモリと第2メモリの間に接続された複数個のストリーム　バッファを有し、前記情報のアドレス及び供給手段は、前記直接マップ　第1キャッシュ　メモリでミスされたデータを供給し、ミスされたデータのアドレスに後続する少くとも1つのアドレスにおけるデータを前記付加的の複数個のストリーム　バ

3

ッファに供給する手段を有してなる請求項17記載のメモリ　システム。

【請求項１９】第１キャッシュ　メモリをアドレスし、このアドレスに応答して第１キャッシュ　メモリ内にミスが生じたか否かを決定し、第１キャッシュ　メモリ内にミスが生じたときは、第２メモリより第１キャッシュ　メモリ及びミスキャッシュに情報を供給することを特徴とするメモリ　アクセス方法。

【請求項２０】第１キャッシュ　メモリと、ミス　キャッシュとに同時に情報アドレスを供給する請求項19記載のメモリ　アクセス方法。

【請求項２１】第１キャッシュ　メモリにミスが生じたときは、第２メモリより第１キャッシュ　メモリ及びミス　キャッシュに対応の情報を供給する請求項19記載のメモリアクセス方法。

【請求項２２】第１キャッシュ　メモリにミスが生じたときは、さらにミス　キャッシュをアドレスし、ミス　キャッシュ内にミスが存しないときはこれに応答してミス　キャッシュより情報を供給し、第１キャッシュ　メモリ及びミス　キャッシュにミスが生じたときは、第２メモリより第１キャッシュ　メモリ及びミス　キャッシュに情報を供給する請求項19記載のメモリ　アクセス方法。

【請求項２３】ミス　キャッシュの１つ並びに第２メモリより移転した第１キャッシュ　メモリよりの情報をミス　キャッシュに供給する請求項19記載のメモリ　アクセス方法。

【請求項２４】パリティ　エラーを検出し、パリティ　エラーの存する場合には、ミス　キャッシュより第１キャッシュ　メモリに対応の情報を供給するステップを含んでなる請求項23記載のメモリ　アクセス方法。

【請求項２５】データを有する情報が付加的にインストラクションの一部を含み、さらに本方法は、第１キャッシュ　メモリ内でミスされたインストラクションを第１キャッシュ　メモリに供給し、ミスされたインストラクションに対するアドレスに後続する少くとも１つのアドレス内の情報を供給するステップを有してなる請求項23記載のメモリ　アクセス方法。

【請求項２６】第１キャッシュ　メモリ内でミスされたデータを第１キャッシュメモリに供給し、ミスされたデータに対するアドレスに後続する少くとも１つのアドレス内のデータを付加的な複数個のストリーム　バッファに供給するステップを含んでなる請求項25記載のメモリ　アクセス方法。

【請求項２７】第１キャッシュ　メモリをアドレスし、このアドレスに応じてミスが発生したか否かを決定し、第１キャッシュ　メモリにミスが発生した場合は、第２メモリより第１キャッシュ　メモリ及びストリーム　バッファに情報を供給するステップを有するメモリ　アクセス方法。

4

【請求項２８】第１キャッシュ　メモリ及びストリーム　バッファに同時に情報アドレスを供給する請求項27記載のメモリ　アクセス方法。

【請求項２９】第１キャッシュ　メモリにミスが生じたときは、第２メモリより第１キャッシュ　メモリ及びストリーム　バッファに対応の情報を供給する請求項27記載のメモリ　アクセス方法。

【請求項３０】第１キャッシュ　メモリ内でミスされた情報を第１キャッシュ　メモリに供給し、かつこのミスされた情報及びミスされた情報のアドレスに後続する少くとも１つのアドレスの情報をストリーム　バッファに供給するステップを含んでなる請求項29記載のメモリ　アクセス方法。

【請求項３１】ミスされた情報がインストラクションを含む請求項30記載のメモリ　アクセス方法。

【請求項３２】第１キャッシュ　メモリ内でミスされたデータを第１キャッシュメモリに供給し、ミスされたデータに対するアドレスに後続する少くとも１つのアドレス内のデータを付加的な複数個のストリーム　バッファに供給するステップを有する請求項31記載のメモリ　アクセス方法。

【請求項３３】第１キャッシュ　メモリ内にミスが発生するとストリーム　バッファをアドレスし、このアドレスに応答してストリーム　バッファ内にミスが存しないときはストリーム　バッファより情報を供給し、第１キャッシュ　メモリ及びストリーム　バッファより情報を供給し、第１キャッシュ　メモリ及びストリーム　バッファ内にミスが生じたときは前記情報を第２メモリより第１キャッシュ　メモリ及びストリーム　バッファに供給するステップを含む請求項27記載のメモリ　アクセス方法。

【発明の詳細な説明】

【０００１】

【技術分野】本発明は、データ処理におけるメモリ　ハイアラーキーの性能（パーフォーマンス）を向上するシステム及び方法に関する。とくに本発明はキャッシュ（CACHE ）メモリのシステム　パーフォーマンス（性能）を向上するシステム及び方法に関する。さらに本発明は、キャッシュ　メモリへのアクセスのミス　ペナルティ（性能低下）を大幅に減少させるシステム及び方法に関する。

【０００２】

【従来の技術】キャッシュ（緩衝記憶）機能は、最近のプロセッサの動作機能に極めて重要な効果をもっているのでますます重要となってきている。表１は、キャッシュ　ミス時間と、マシンのパーフォーマンスのミスの影響を示すものである。最近の10年間に、主メモリのアクセス時間よりもサイクル　タイムが遙に急速に減少している。インストラクション当りのマシン　サイクルの平均数も顕著に減少しており、とくにコンプレックス（総

5

合）　セット　コンピュータ（CISC）　マシンよりレジュースト（減少）　インストラクション　セット　コンピュータ（RISC）マシンへの転移（transition）が含まれる場合、これが著しい。これら２つの影響は相乗的であり、ミス動作　コストを極めて増大させる結果を招来する。例えば、ディジタル　イクイップメント　コーポレイション VAX 11/780 のキャッシュ　ミスは、平均インストラクション　エクゼキューション（実行）の60％を要するのみである。従ってインストラクションがキャッシュ　ミスを有していても、そのパーフォーマンスは60％のスロー　ダウンを示すのみである。しかしディジタル　イクイップメント　コーポレイション　WRL の如き RISC マシンがミスを有すると、そのコストはほぼ10倍のインストラクション　タイムとなる。とくにメモリ　アクセス　タイムとマシン　サイクル　タイムの比が＊

6

＊増加すると、この傾向は継続する。将来、サイクル当りの２つのインストラクションを遂行する超大形マシンで主メモリへ至るすべての通路のキャッシュ　ミスは、100 インストラクション　タイム以上のコストを要することとなる。良く知られているキャッシュ　デザイン技術を注意深く応用した場合でも、100 インストラクション　タイム以上の主メモリ回転待時間（latancy ）を有するマシンでは、メモリハイアラーキーの固有性能（パーフォーマンス）の半分以上を直に失い易いこととなる。このことから最新のメモリ　ハイーラーキーのハード　ウエア及びソフト　ウエアの双方の研究が極めて重要となってきている。

【０００３】

【表１】

Machine	cycles per instr	cycle time (ns)	mem time (ns)	miss cost (cycles)	miss cost (ns
VAX 11/780	10.0	200	1200	6	
WRL Titan	1.4	45	540	12	
?	0.5	4	280	70	1

【０００４】

【発明の開示】本発明は、一般に第１キャッシュ　メモリと第２メモリを有するメモリ　システムに関する。これらの第１キャッシュ　メモリと第２メモリの間にミス　キャッシュを接続する。ミス　キャッシュは、第１キャッシュ　メモリよりも小さなメモリ容量を有する。情報のアドレスを行うこれらの第１キャッシュ　メモリ、第２メモリ及びミス　キャッシュにある特定の手段を接続し、第１キャッシュメモリ内にアドレス情報のミスが発生した際、情報をアドレスし、第１キャッシュ　メモリ及びミス　キャッシュへの情報の供給を行う。

【０００５】本発明によるメモリのアクセス方法は、第１キャッシュ　メモリをアドレスし、このアドレスに応答する第１キャッシュ　メモリ内にミスが発生するか否かを確める。第１キャッシュ　メモリ内にミスが発生したときは、第２メモリより第１キャッシュ　メモリへ、並びにミス　キャッシュに情報が供給される。

【０００６】本発明はさらに、第１キャッシュ　メモリ、第２メモリ並びに付属のミス　キャッシュで、第１キャッシュ　メモリよりも小容量を有し、第１キャッシュ　メモリと第２メモリ間に接続されているミス　キャッシュを有し、さらにこれらの第１キャッシュ　メモリ、第２メモリ及び付属のミス　キャッシュに接続されており、第１キャッシュ　メモリ内に情報のアドレスのミスが生じたときは、第１キャッシュ　メモリ及び前記付属のミス　キャッシュに情報を供給する手段を具えたメモリ　システムに関する。

【０００７】

【実施例】以下図面を参照して本発明を説明する。図１は、メモリ　システムの本発明に関係ある部分のみを示す。同一のチップ22上に、あるいは最新のパッケージ技術によって製造された単一の高速モジュール上に、CPU（中央処理ユニット）12、フローティング　ポイント　ユニット（FPU ）14、メモリ　マネージメント　ユニット（MMU ）16（例えばトランスレーション　ルック・アサイド　バッファあるいはTLB ）、及び第１レベル　インストラクション及びデータ　キャッシュ18及び20を設ける。（以下本明細書においては、セントラル　プロセッサ24をシングル　チップと称するが、これはチップまたはモジュールを意味する。）このチップ22のサイクル時間は、インストラクションの発出速度（レート）よりも３～８倍長い。（すなわち、３～８インストラクションによって１つのオフ・チップ　クロック　サイクルとなる。）これは、サイクル当り多くのインストラクションを発出すること（スーパー　カレンダまたは極めて長いインストラクション　ロード VLIW ）により、極めて速いオン・チップ　クロックを設けるか（すなわち、通称スーパー　パイプライニング）、あるいは、システムの残部に対するよりもプロセッサ　チップ22に対し高速技術を用いる（例えば、 GaAs 対 BiCMOS ）ことによって達成できる。

【０００８】オン・チップ　キュッシュ18及び20の予期されるサイズ（寸法）は、プロセッサ24の製造技術によって異なるが、高速技術は一般に小サイズのオン・チップキャッシュを要することとなる。例えば極めて大きなオン・チップ　キャッシュ18及び20がCMOSに用いるに適しているが、GaAsまたはバイポーラによるプロセッサには近接限界（near term ）より小形のキャッシュ18及び

7

20しか使用できない。従ってGaAs及びバイポーラはより急速な動作が可能であるが、これらの小形のキャッシュよりの高いミス　レートによって、GaAsまたはバイポーラ　マシンと、稠密なCMOS　マシン間の実際のシステム　パフォーマンス比は、これらのゲート速度の比以下となる。これらすべての場合、第１レベル　キャッシュ18及び20は、直接マップされるものとした。これはもっとも早いアクセス時間を得られるからである。オン・チップ　キャッシュ18及び20のライン寸法は、大体16B ～32B の範囲である。データ　キャッシュ20は、ライト・スルーまたはライト・バックの何れでも良い。本明細書ではこれらの長・短は論じない。

【０００９】第２レベル　キャッシュ26は、512KB ～16MBの容量とし、超高速の静（スタティック）RAM より構成される。これは、第１レベル　キャッシュ18, 20と同じ理由で、直接マップされる。この大きさのキャッシュ26のアクセス　タイムは16～30ns程度である。これはこのキャッシュに、４～30のインストラクションのアクセス　タイムを与える。このキャッシュ26のアクセスタイムに比較して、プロセッサ24の相対速度は、第２レベル　キャッシュはパイプライン（制御）されるを要し、これによって充分な帯域幅を得るようにするを要する。例えば第１レベル　キャッシュ18, 20がライトスルー　キャッシュである場合を考える。各６または７インストラクション毎に１つの平均速度でストア（蓄積）が生ずるのが典型的であるため、パイプラインされていない外部キャッシュ26は７インストラクション　タイムより大なるアクセス　タイムに対してはトラフィックの記憶を取扱うに充分な帯域幅を有していない。これ迄何年もの間、キャッシュ　メモリは主フレーム内で、パイプライン制御されてきていた。しかしこれはワーク　ステーションの最近の開発によるものである。最近ECL I/O 及びレジスタまたは入力及び出力にラッチ28を有するキャッシュ　チップが開発されている。これらはパイプラインされたキャッシュとして理想的なものである。第２レベル　キャッシュ内のパイプライン数は２～３として良く、この数は、プロセッサ　チップ22よりキャッシュ　チップに至るパイプ段28、及びキャッシュ　チップよりプロセッサ24に戻るパイプ段がフル（完全）または半（ハーフ）パイプ段であるかによって定まる。

【００１０】この速度（例えばMIP 当り数メガ　バイト）のプロセッサ24に充分なメモリを与えるためには、主メモリ34は512MB －4GB の範囲であるを要する。これは約1000個のドラムを含む16Mbのドラム（DRAM）を使用することをも意味する。主メモリ　システム34は第２レベル　キャッシュのアクセスに比し、大体約10倍の長いアクセス　タイムを要する。かかる主メモリ34へのアクセスタイムは、多くのカードに分散している数千のドラム（DRAM）中のアドレス及びデータ信号を抽出（ファンアウト）ずるに要する時間がその主要部を占る。この

8

ためより高速なドラムが出現しても、主メモリへのアクセス　タイムは大体同じ程度に留まる。主メモリへのアクセス　タイムが大であることは、第２レベル　キャッシュのライン　サイズを128 または256Bとするを要することとなる。例えば16B のみの場合は、320ns 後に再回（リターン）することとなる。これは50MB/sec のバスのバンド幅となる。このバス　バンド幅の10MIP プロセッサは、１つのメモリ位置より他のメモリ位置へコピーをするとき、バス　バンド幅の制約を受けるので、100 ～1000のMIP プロセッサを使用してもその能力の追加は僅かである。これはプロセッサのシステム性能（パーフォーマンス）を考える上で極めて重要な点である。

【００１１】このベース　ライン　システム10に関しいくつかの観察を行うと次の如くである。第１にこのシステムのメモリ　ハイアラーキーは、例えばVAX 11/780のマシンと極めてよく類似しており、ハイアラーキーの各レベルのみが１レベルCPU12に向って上に移動する。例えばVAX 11/780 の8KB のボードレベル　キャッシュはオン・チップ移動をする。早期のVAX モデルの512KB －16MB主メモリはボードレベル　キャッシュ26となっていた。780 型の主メモリがはじめて入力トランスファー寸法が大となった（ここでは128－256B　対 VAXの512 ページ）。この方式（システム）の主メモリ34は、早期の780 型のディスク　サブシステムの寸法と同じで、ページング及びファイル　システム　キャッシングの如き同じ機能を行う。

【００１２】システム10のメモリ　ハイアラーキー　パーフォーマンスの各段階にパラメータを付したものを図２に示した。このシステムに見積られた実際のパラメータは次の如くである。インストラクション　イッシュウ　レート 1,000MIP 、ファースト　レベル　インストラクション　4KB、４KBデータ　キャッシュ 16B　ライン付、1MB 第２レベル　キャッシュ 128B ライン付。ミス　ペナルティは、第１レベルに対し、24インストラクション　タイムであり、第２レベルに対し320 インストラクション　タイムである。テスト　プログラムの特性を表２に示す。これらのベンチマーク（水準）は、現在使用されている多くのトレース（計画）に比し、かなり長いものである。しかしこの作業にはマルチプロセシングの効果はモデルされていない。

【００１３】図２に見られるように、殆どのベンチマークは、第１レベル　キャッシュのミスで潜在性能（パーフォーマンス）の半分以上を失っている。第２レベル　キャッシュ　ミスによっては極く少いパーフォーマンスが失われるのみでる。これは主として、実行プログラム　サイズに比較して第２レベル　キャッシュのサイズが大であることによるものでる。大量のプログラムによるより長いトレースは、極めて重要な第２レベル　キャッシュのミスを生ずる。本説明に用いる試験例は、第２レベル　キャッシュ機能の有意性に比し極めて少いので、

9　　10

第２レベル　キャッシュ　ミスの詳細は検査しない。

【表２】

program name	dynamic instr.	data refs	total refs.	program type
cccm	31.5M	14.0M	45.5M	C compiler
grr	134.2M	59.2M	193.4M	PC board CAD t
yacc	51.0M	16.7M	67.7M	Unix utility
met	99.4M	50.3M	149.7M	PC board CAD t
linpack	144.8M	40.7M	185.5M	numeric, 100x1
livar	23.6M	7.4M	31.0M	LFK (numeric loops)

【００１４】想定されるパラメータは所定範囲の極端のもの（最大パーフォーマンス　プロセッサに最小サイズのキャッシュ）であるため、他の形態はメモリ　ハイアラーキーのパーフォーマンスにおける対応のロスは、これより少いものとなる。それにも拘らず、興味ある範囲内での任意の形態は、メモリ　ハイアラーキーの潜在パーフォーマンスのかなりの比率部分を失う。これはシステムのパーフォーマンスの中の最大の部分は、メモリ　ハイアラーキー　パーフォーマンスの改良によって得られ、これはCPU のパーフォーマンスをより増加しようとする試みによって得られるものではないことを意味する。（後者は例えば、インストラクションの並列発生をより増加させることにより。）以下本明細書で述べる主課題は、低価格で、ベースライン　メモリ　ハイアラーキーのパーフォーマンスを改良向上させることでる。最後に付け加えると、CPU コア（CPU12，EPU14，MMU16，第１レベル　キャッシュ18及び20を含む）のパーフォーマンスのコンプロマイズ（妥協）を避けるため、研究すべき本技術に必要なすべての付加的ハードウエアはCPU コアの外側（すなわち、第１レベル　キャッシュ18及び20の下側）でなければならない。かくすることにより、これらの付加的ハードウエアはキャッシュ　ミスの際にのみ関与することとなり、通常のインストラクションの実行に絶対に（クリティカルに）必要な通路内には存しなくなる。

【００１５】キャッシュのミスは４つのカテゴリーに分類することができ、これらは次の如くである。

コンフリクト（抵触）、……conflict

コンパルソリ（強制）、……compulsory

キャパシティ（容量）、……capacity

コヒーレンス（凝集）、……coherence

コンフリクト　ミスとは、キャッシュが完全に付属的（アソシアティブ）であり、少くとも最長時間未使用LRU （リースト　リセントリィ　ユースト）の入換え（リプレースメント）を有しているときには起らないミスである。コンパルソリ　ミスとは、キャッシュが第１にデータの一部の最初のものであるため、すべてのキャッシュ構成（オーガニゼーション）に必然的なミスである。キャパシティ　ミスとは、キャッシュのサイズが参照間のデータを保持するに充分でないときに生ずる。コーレンス　ミスとは、マルチプロセッサ　キャッシュ　コンシステンシィ（確実性）の保存を無効にするとき以外には生じないミスである。

【００１６】直接マップされたキャッシュは、これらが付属性（associativity ）に欠けているためより多くのコンフリクト　ミスを有するが、これらに対するアクセスタイム　コストを考慮するとき、これらのパーフォーマンスはセット・アソシアティブ　キャッシュよりも良い。実際上、究極的通路が時間を要するRAM のアクセスである場合に、この直接マップ　キャッシュのみが唯一のキャッシュ形態となる。コンフリクト　ミスは、すべての直接マップ　キャッシュ　ミスの20％～40％の間にあることが一般である。図３は、試験モデルに対するコンフリクトによりミスのパーセントを示すものである。第１レベル　データ　キャッシュ　ミスの平均で30％は、コンフリクトによるものであり、また第１レベル　インストラクション　キャッシュ　ミスの20％はコンフリクトによるものである。これらはかなり大きな比率であるため、直接マップ　キャッシュ　メモリに限界的なアクセス通路を付加することなく、或る程度の付加的付属性（associativity ）を与えることにより“我々の成果（ケーキ）を享受し、さらにその恩恵にあずかる（食べる）”ことが好適である。

【００１７】図４のミス　キャッシュ　システム40に示す如く、チップ上の第１レベル　キャッシュ20と第２レベル　キャッシュへのアクセス　ポートの間に小形のミスキャシュ42を配置することによって直接マップ　キャッシュ20の対応性を付加することができる。ミス　キャッシュ42は、データの２～８程度のキャッシュ線44を有する小形の完全・アソシエティブ　キャッシュである。ミスが発生すると、データは直接マップ　キャッシュに戻されるのみでなく、ミス　キャッシュ42にも戻され、ここにおいて、LRU （最長時間未使用）アイテムを入換える。上位キャッシュ20が検算（prove ）される毎に、ミス　キャッシュ42も検算される。上位キャッシュ20内にミスが生じ、アドレスがミス　キャッシュ42に一致しているときは、直接マップ　キャッシュ20は次のサイクルでミス　キャッシュ42より再ロードされる。かくすると、長期のオフ・チップ　ペナルティが、短期のオンサイクル　オン・チップ　ミスで置換えられる。この配置は、クリティカルな通路が悪化されないという要求を満足する。その理由は、ミス　キャッシュ自体はプロセッサの実行の通常のクリティカル通路に存しないからである。

11

【００１８】コンフリクト　ミスの除去において、種々のミス　キャッシュ構成の成功率を図５に示す。第１に着目すべきことは、ミス　キャッシュ42によって、インストラクション　コンフリクト　ミスよりも、より多くのデータ　コンフリクト　ミスが除かれることである。これは次の如くして説明できる。ほとんどすべての場合にあてはまるように、キャッシュ　サイズよりもプロセジュア（手続）のサイズが小である限り、１つのプロセジュア内の各インストラクションは互にコンフリクトを生じないように、インストラクション　コンフリクトは広く拡がる傾向がある。従ってインストラクション　コンフリクト　ミスは他のプロセジュアが呼出されるときに起り易い。ターゲット　プロセジュアは、呼出し（calling）プロセジュアに関し、すべての個所にマップされるので、極めて多くのオーバーラップを来す。各プロセジュアにおいて、60の異なるインストラクションが実行されるとすると、試験される最大寸法のミス　キャッシュ42内で、コンフリクトミスは15ライン以上に分離される。換言すると、小さなミス　キャッシュ42は、全オーバー　ラップを収容しきれず、これを使用しうるようにする前に反復して再ロード（reload）する必要がある。この型式の参照パターンは最悪のミスキャッシュ　パーフォーマンスを示す。

【００１９】一方、データ　コンフリクトは極めて接近した間隔で生じうる。２つのキャラクタ　ストリングを比較する場合を考える。２つのストリングの比較点が同じラインにマップするとすると、互に相異なるストリングへの交互の参照（リファレンス）は常にキャッシュ内で失われる。この場合２つのエントリーのみのミスキャッシュ42によりこれらのコンフリクト　ミスのすべてが除かれる。これは他方のパーフォーマンスの極端な場合であることは明らかであり、図５の結果は関連のプログラムに基づくパーフォーマンスの範囲を示している。それにも拘わらず4KB データ　キャッシュに対して、２つのエントリー44のみのミス　キャッシュ42によって、データ　キャッシュ　コンフリクト　ミスの平均25％、または全データ　キャッシュ　ミス（図６）の13％が除かれる。ミスキャッシュ42を４つのエントリー44に増加させると、38％のコンフリクト　ミスが除去しうるか、全体で36％のデータ　キャッシュ　ミスが除かれる。４エントリー44のミス　キャッシュエントリーによる改良は僅かであり、15エントリーの44を設けたときも、データ　キャッシュの全体の減少は、25％増加したのみであった。

【００２０】データ　キャッシュ20のサイズを倍増させることによってミスが32％減少するので、（データ　キャッシュ　サイズを4Kより8Kに増加させると、この指標値以上となる）、第１レベル　キャッシュ20内の各追加ラインは、約0.13％だけミスの数を減少させる。ミス　キャッシュ42は、記憶ビット当りにして、データ　キャッシュ20よりもより多くの面積を必要とするが、２ライン　ミス　キャッシュ42内の各ラインは、ミス率において50倍の大きさのマージンの改良を行うので、レイアウトのサイズの差を補って余りがある。

12

【００２１】図５と図３とを対比すると、コンフリクトによるミスのパーセントが高い程、ミス　キャッシュ42がより有効にこれらのミスを消去することが判る。例えば、図３では、“met”は、全体のデータ　キャッシュ20のミスに対し最高のコンフリクト　ミスを示している。同様に“grr”及び“yacc”は、コンフリクト　ミスの平均パーセントよりも大きな比率を示している。ミス　キャッシュ42はこれらのプログラムに対しても極めて大きな貢献をする。“linpack”と“ccom”はコンフリクト　ミスの最小パーセント値を示し、ミス　キャッシュ42はすべてのプログラムのこれらのミスのごく数パーセントを取除く。これはプログラム　データ　コンフリクト　ミスに大きな比率を有しているときは、全体の密度の理由で、これらを或る程度粉砕（cluster）すべきことを示している。これはプログラムが、少数のコンフリクト　ミス、例えばミス　キャッシュより、給付（benefit）される“liver”の如きミスを含むことを妨げない。しかしコンフリクトミスのパーセントが増加するにつれ、ミス　キャッシュによるこれらのミスの除去率も増加する。

【００２２】直接マップ　キャッシュとミス　キャッシュを有するシステムについて考える。ミスが生ずると、データは、ミス　キャッシュと直接マップ　キャッシュの双方にロードされる。ある意味では、このデータの二重操作は、ミス　キャッシュ内の記憶蓄積スペースを浪費する。ミス　キャッシュ内の二重化されるアイテム数は、１（ミス　キャッシュ　マップ内のすべてのアイテムが直接マップ　キャッシュ内のラインと同じになっている場合）より、エントリーの全数（ミス　キャッシュをヒットしない一連のミスが生ずる場合）の範囲に亘る。

【００２３】ミス　キャッシュ42をより良好に使用するため、図７のキャッシュ　システム50内に示す小形の完全付属（アソシエティブ）ミス　キャッシュ52に対する別の置換アルゴリズムを使用すごとができる。ミスにより要求されるデータをミスキャッシュ42内にロードする代りに、54に示す如く、直接マップ　キッシュ20よりビクティム　ラインによって完全付属ミス　キャッシュ52にロードすることができる。これを“ビィクティム　キャッシング（犠牲キャッシング）”と称する。ビィクティム　キャッシングには、データ　キャッシュ20にも、ビィクティムキャッシュ52にも何れにもデータ　ラインは現れない。これは、ビィクティムキャッシュ52は、直接マップ　キャッシュ20より放出されるアイテムのみがロードされるからである。ビィクティム　キャッシュ52をヒットするような直接マップ　キャッシュ20内のミス

13

の場合には、直接マップ　キャッシュ　ラインとマッチするビクティム　キャッシュ　ライン56の内容をスワップ（交換）する。

【００２４】参照ストリームによって、ビクティム　キャッシュはミス　キャッシュに小さなあるいは極めて重大な改良の何れかを行うことができる。この利点はミスキャッシュ内の二重化に対応して定まる。ビクティム　キャッシュは常にミスキャッシュの改良を行う。

【００２５】１例として、コール　サイトとコンフリクトを生ずる内側ループ内の小手続き（プロセジュア）を呼出すインストラクション参照ストリームを考える。このプログラムのエクゼキーション（実行）は、ミス　キャッシュ42内の位置数よりも大きなコンフリクト　ループに沿ったパス長を有することもある。この場合、ミス　キャッシュは価値を生じないこともある。その理由は、キャッシュ内に常時二重化アイテムがフラッシュされるからである。しかし、これに代えてビクティム　キャッシュを使用すると、捕捉しうるパス長はほぼ２倍となる。これは１組のコンフリクト　インストラクションが直接マップ　キャッシュ20内で生きており、また他の１組がビクティム　キャッシュ52内で生きているからである。このループに沿ってエクゼキュションが進行するにつれて、これらのアイテムのトレードが行われる。

【００２６】ビクティム　キャッシングによって除かれるコンフリクト　ミスのパーセントを図８に示した。１つのライン56で構成されているビクティムキャッシュ52は、必ず２つのライン44で構成するを要するミス　キャッシュ42と逆に有用でさえある。ミス　キャッシュと比較してすべてのベンチマークは改善されている。しかしベンチマークのインストラクション キャッシュ18のパーフォーマンス及びデータ　キャッシュ20のパーフォーマンスは長い連続参照ストリーム（例えば、“ccom”及び“リンパック”）がもっとも改良されている。

【００２７】図９は、ビクティム　キャッシュによって得られたと思われるミスの全体の減少率を示す。図６と図９との比較より判るように、ビクティム　キャッシュ52のパーフォーマンスは、２倍のエントリー数を有しているミス　キャッシュ42よりも場合によって良好である。例えば“yacc”のデータ　キャッシュ20のパーフォーマンスを、１・エントリービクティム　キャッシュ52及び２・エントリー　ミス　キャッシュ42で考えて見る。ビクティム　キャッシュ52はビクティムを放棄しないので、状況によってビクティム　キャッシュはエントリー数が２倍のミス　キャッシュよりもミスの数が少くなる。例えば、新規なデータの評価に多くのキャッシュ　ミスが生ずる（例えば、強制（compulsory）ミス）ことを考えると、ミス　キャッシュ42とビクティム　キャッシュ52の双方をフラッシュ　アウトすることが有効である。次で他の新しいラインが参照される場

14

合を考えると、ミス　キャッシュ42を有するシステム40と、ビクティム　キャッシュ52を有するシステム50との双方にミスが生じている。ラインの古い内容が次に参照されると、ミス　キャッシュ42はこのアイテムを保有していないが、ビクティム　キャッシュ52はこれを保有している。従ってミス　キャッシュ42を有するシステム40は、メモリ　ハイアラーキーの次位のレベルで２つのミスを有するが、ビクティム　キャッシュ52を有するシステム50は１つのみのミスを有することとなる。

【００２８】図10は、種々のサイズの直接マップ　データ　キャッシュ20をバックアップするときの、１，２，４，15エントリーのビクティム　キャッシュのパーフォーマンスを示す。一般に小さい方の数の直接マップ　キャッシュ20が、ビクティム　キャッシュ52の付加によって、より多くの利益を得る。参考のため、各キャッシュ　サイズに対するコンフリクト　ミスの全パーセントを示してある。ビクティム　キャッシュ52のパーフォーマンス対直接マップ　キャッシュ20のパーフォーマンスには２つのファクターが存する。その第１は、直接マップ　キャッシュ20のサイズが大となるにつれ、ビクティム　キャッシュ52の相関的サイズは小となることである。直接マップ　キャッシュ20が大となり、ライン　サイズを小（16B）としている為、ビクティム　キャッシュにより容易に除きうる稠密なマッピングのコンフリクトの可能性が減少する。第２には１KBより32KBへ向ってコンフリクト　ミスは僅か減少する。既述の如く、コンフリクトミスのパーセントが減少するにつれ、ビクティム　キャッシュ52によって除かれるミスのパーセントも減少する。しかし極めて大きなキャッシュに対しては、コンフリクト　ミスのパーセントが増加するため、第１の効果がより大きな影響を及ぼし、ビクティム　キャッシュ52のパーフォーマンスは僅かしか増加しない。

【００２９】図11は、種々のライン　サイズの４KB直接マップ　データ　キャッシュ20に対するビクティム　キャッシュ52のパーフォーマンスを示す。予期したように、このレベルでライン　サイズが増加するにつれて、コンフリクト　ミスの数も増加する。コンフリクト　ミスのパーセントが増加すると、ビクティム　キャッシュ52によって除去されるかかるミスのパーセントも増加する。ビクティムキャッシュ52を有するシステム50は、長いライン　サイズにおいて、ビクティム　キャッシュを設けないものよりも利益を得る。その理由は、ビクティムキャッシュ52は、長いキャッシュ　ラインより由来したコンフリクトによるミスを除く助けをするからである。ビクティム　キャッシュ52内にデータ記憶のために用いられる面積が一定に保持されるとしても（例えば、ライン　サイズが倍となると、エントリー数は半分にカットされる）、ライン　サイズが増加したとき、ビクティム　キャッシュ52のパーフォーマンスは

15

より改良されるか、少くとも同じ状態を保つ。

【００３０】キャッシュのサイズが増加すると、ミスの大きなパーセントは、コンフリクト及びコンパルソリー　ミスとなり、キャパシティ　ミスの比率は減少する。（キャッシュが全プログラムよりも大きい場合は、コンパルソリー　ミスのみが残るので、当然これを除く。）従ってビクティム　キャッシュ52は、第２レベルキャッシュ26にも有効であると期待される。ライン　サイズの増加とともに、コンフリクト　ミスの数が増加するため、第２レベルキャッシュ26のライン　サイズが大となると、ビクティム　キャッシュ52の潜在的有用性が増加する傾向にある。第１レベル　キャッシュ20の場合と同様に、第２レベル　キャッシュ26内でコンフリクト　ミスのパーセントが大となるにつれて、ビクティム　キャッシュ52によって除かれるコンフリクト　ミスのパーセントが大となる。

【００３１】ビクティム　キャッシュの興味ある一面は、キャッシュのハイアラーキーにおけるインクルージョン　プロパーティ（算入特性）を冒涜（violate ）することである。これは、マルチプロセッサのキャッシュのコンシステンシィ（一貫性）のアルゴリズムに悪影響を及ぼすだけでなく、キャッシュ　シミュレーションに用いられるアルゴリズムにも影響を及ぼす。例えば、所定のCPU 基準ストリーム上の第２レベル　キャッシュ26に対するミスの数は、その頂上（トップ）にある第１レベル　キャッシュ26のサイズ（但し、ライン　サイズではない）に無関係である。しかしながらビクティム　キャッシュ52は、第１レベルのみでなく、第２レベルにおいてもコンフリクトする多くのラインを含有することができる。このため、第１レベル　ビクティム　キャッシュ52の使用は、第２レベルにおけるコンフリクト　ミスの数も減少させることができる。第２レベル　キャッシュ26に対するビクティム　キャッシュ52の研究に当り、第１レベル　ビクティム　キャッシュ52を設ける場合と、設けない場合との両方の形態を考える必要がある。

【００３２】メガ　バイトの第２レベル　キャッシュ26に対するビクティム　キャッシュ52の念入りな調査には、数百万のインストラクションのトレースが必要となる。現在我々はより小さいテスト　ケースに対するビクティム　キャッシュのパフォーマンスのみを有しており、多メガ　バイトの第２レベル　キャッシュ26に対するビクティム　キャッシュのパフォーマンスを得ることについては目下取進め中である。

【００３３】とくに第１レベルのオン・チップにおけるミス　キャッシュ42のその他の重要な使途は、イールド（歩留り）の向上である。すべてのインストラクション及びデータ　キャッシュ18及び20のバイトにパリティが維持されており、データ　キャッシュ20がライト・スルーである場合、キャッシュ　パリティ　エラーはミスとして取扱うことができる。レフィル　パスがキャッシュをバイパスするときは、このスキムは、ハードのエラーを有するチップを使用することを許容する。（実際上、バイトのパリティに対しては、１バイト当り、最大で１つの悪いビットがあるとすると、キャッシュ内のすべてのビットの1/9 迄が誤っていることもありうる。）残念乍ら、ミス　キャッシュ42が存せず、linpack （例えば、saxpy）の内側ループが欠陥のあるライン上にランドするか、構造変数に用いる周波数が欠陥ライン上のものである場合は、システムのパフォーマンスは大幅に低下（degrade ）する。（例えば、あるコード　セグメントで係数４より大にデグレードする）。さらに、欠陥の位置によって、ランダムの様相でパフォーマンスのデグレードがチップ毎に変化する。これは、プロジェクトのエンジニアリングの開発における潜在性イールドの強化を制限する。しかしミス　キャッシュ42を付加することによって、欠陥を引起すパリティ　ミスのペナルティは１サイクルのみとなり、これはオフ・チップ　ミスよりもマシンのパフォーマンスに与えるインパクトは遙に小となる。従って欠陥の数が小であり、ミス　キャッシュ42で充分取扱えるものであれば、ハードの欠陥のあるチップを生産系で使用することができる。もし、ミス　キャッシュ42を、生産上の欠陥のあるシステムのパフォーマンスの改良に使用すべきときは、インストラクション　ミス　キャッシュあるいは単に１つのエントリーのみを有するミス　キャッシュも有用である。

16

【００３４】前に述べたビクティム　キャッシュ52はパリティ　エラーに起因するミスの修正には有用ではない。これはビクティムがパリティ　エラーでコラプス（退化）しており、セーブに価しないからである。しかしビクティム　キャッシュ52は次の如くの変化を加えることによってエラー修正にも使用することができる。キャッシュ　ミスがパリティ　エラーによって生ずる場合には、ビクティムキャッシュ52に入力（ミス）データをロードし、ビクティムをロードしない。かくすると、通常のミスに対するビクティム　キャッシュ52のように動作し、ミス　キャッシュ42はパリティ　ミスに対し動作する。このような僅かな変形によって、ミス　キャッシュ42がエラーのレカバリーに用いられ、ビクティムキャッシュのより良好なパフォーマンスを組合せることができる。

【００３５】コンパルソリー（強制）ミスは、如何なるキャッシュ構成でも、一部のデータに最初に参照されるために必要とされるミスである。キャパシティ　ミスは、キャッシュの大きさが、参照の間にデータを充分保持するものでないときに生ずるミスである。キャパシティ　ミス及びコンパルソリー　ミスを減少させる１つの方法は、長いキャッシュ　ライン　サイズまたはプレフェッチ方法の如きプレフェッチ技術を用いることである。しかしながら、ミスのレートを増加させることなし

17

に、また転送すべきデータの量を過に増加させることなしにライン　サイズを任意に大とすることはできない。本章では、長いライン及び過剰のプレフェッチに関する従来の問題を減少させ乍ら、キャパシティ　ミス及びコンパルソリーミスを減ずる技術を研究することとする。

【００３６】ライン　サイズが長くなると、各種の異なるプログラム及びアクセス　パターンに対し固定した転送（トランスファー）サイズを設ける点で不利益を生ずる。プレフェッチ技術は、プログラムの実際のアクセス　パターンにより良く適合しているので興味がある。このことは、インストラクション　ストリームまたは、ユニット　ストライド　アレイ　アクセスの如き、長い準連続（quasi-sequential）アクセス　パターンのパフォーマンスの改良にとくに重要である。

【００３７】３つのプレフェッチ　アルゴリズムの詳細な解析が、スミス　アラン　ジュニア（Smith Alan. J.）により“Cache Memorie ”として、Computing Servеys 1982, 9, pp473－530 に発表されている。プレフェッチでは、各参照後常にプレフェッチを行う。これは我々のベース　システム10では実際的でない。その理由は、単一のレベル・２キャッシュ　レファレンス（参照）を必要とする時間内に数多くのレベル・１　キャッシュ　アクセスが行われるからである。これはインストラクション　キャッシュ18より、１サイクル当り複数のインストラクションをフェッチし、同時に１サイクル当りデータ　キャッシュ20にロードまたは蓄積を行うマシンにおいて特に事実である。プレフェッチ・オン　ミス（ミスの際のプレフェッチ）及びタグド・プレフェッチ（タグ付プレフェッチ）はより有望な技術である。プレフェッチ・オン　ミスにおいては、ミスが生ずると、次のラインもプレフェッチする。この技術では、純粋シーケンシャル　レファレンス　ストリームに対するミスの数を半分にカットする。タグド・プレフェッチは、さらにこれより良好である。この技術においては、各ブロックは付随するタグ　ビットを有する。あるブロックがプレフェッチされると、そのタグ　ビットは“０”（ゼロ）にセットされる。このブロックが使用される毎にタグ　ビットは１にセットされる。このブロックが０より１に転移する毎にその後続のブロックがプレフェッチされる。フェッチが充分急速に行われると、これにより純粋シーケンシャル　リファレンス　ストリーム内のミスの数は０に減少する。残念乍ら、ベースシステム10では、レーテンシィ（潜在性）が極めて大であるためこれは不可能である。図12は、Ｃ　コンパイラー　ベンチマークの遂行中に必要とするプレフェッチ　ラインに至る迄の時間（インストラクション　発出迄の時間）を示すものである。当然のことながら、ライン　サイズは４インストラクション分であるため、マシンをキャッシュされないストレート・ライン　コードで維持するためには、プレフェッチ　ラインは、４インストラクション時

18

間内に受信されなければならない。ベース　システムでは、第２・レベル　キャッシュ26は、アクセスに多くのサイクルを要し、かつマシンは各サイクル当り実際に数多くのインストラクションを発出するため、タグド　プレフェッチのみが、所要のインストラクションを設けるのに、ワン・サイクル・アウト・オブ　メニー　ヘッド　スタートを有する。

【００３８】本発明で必要なことは、タグ転移の生ずる前にプレフェッチをスタートさせることである。図13に示したシステム60におけるストリーム　バッファ62と称されるメカニズム（機構）によりこれを行うことができる。ストリーム　バッファ62は、それぞれタグ66より成る一連のエントリー64、利用可能ビット68及びデータライン70より成る。

【００３９】ミスが発生すると、ストリーム　バッファ62は、ミス　ターゲットより出発して、順次のラインのプレフェッチを開始する。各プレフェッチリクエストが送出されると、このアドレスに対するタグ66がストリーム　バッファ62に入力され、ここで利用できるビット68は誤り（false ）としてセットされる。プレフェッチデータ70が戻ってくると、そのタグ66と共にエントリー64内に収納され、ここで得られるビット68は真値（true）とされる。ミスによってリクエストされる順次のラインは、キャッシュ20でなくバッファ62に収納される。かくすることによって、必要でないデータによってキャッシュ20が汚染されることが防止される。

【００４０】次々のキャッシュへのアクセスにおいても、これらのアドレスをストリームバッファ62に記憶されている第１アイテムと比較する。この参照において、キャッシュ20ではミスし、ストリーム　バッファ62ではヒットしたとすると、キャシュ20はストリーム　バッファ62より単一サイクルで再ロードすることが可能である。これはオフ・チップ　ミス　ペナルティよりも遙に急速である。ここにおけるストリーム　バッファ62は簡単なFIFO　キュー（待合せ）と考えられ、キューの初頭のもののみが、タグ比較器72を有しており、ストリーム　バッファ62より移動されたエレメントは、如何なるラインをもスキップすることなしに順次厳密に移動させることを要する。この簡単なモデルでは、非順番（non-sequential）ライン　ミスは、キューの次の下位に要求されるラインが既に出現していても、ストリーム　バッファ62をフラッシュさせ、ミス　アドレスより再スタートさせる。以下に、既にフェッチしたラインを順番（シーケンス）外としうるより複雑なストリーム　バッファについて述べる。

【００４１】ライン64がストリーム　バッファ62よりキャッシュ20に移されると、ストリーム　バッファ62内のエントリーは１つだけシフトし、新規な後続のアドレスがフェッチされる。次の後位アドレスはインクレメンタ74によって発生される。第２レベル　キャッシュ26への

19

パイプ ライン インタフェイスによって、ストリーム バッファ62は第２レベル キャッシュ26の最大帯域幅に充填され、プロセスにあたり、多くのキャッシュ ラインを同時にフェッチすることができる。例えば、インストラクション キャッシュ18のミスの16B ラインの再充足のレーテンシィ（回転待ち時間…latancy ）が12サイクルと見なす。パイプ ラインされたメモリ インタフェイスで、各４サイクル毎に新しいライン リクエストを受入れられるものを考える。４・エントリー ストリーム バッファ62は、常時３つのリクエストが待機している状態で、サイクル当り１つの速度で４Ｂのインストラクションを与えることができる。従って順次のインストラクション実行中、長いレーテンシィ キャッシュ ミスは生じない。これは、同時には１ラインのみしかプレフェッチされない純粋順次参照ストリームにおけるタグ付プレフェッチのパフォーマンスと違っている。ここでは順次のインストラクションは、各３サイクルに１つのインストラクションに等しい帯域幅（すなわち、12サイクル レーテンシィ／４インストラクション 各ライン当り）でのみ供給される。

【００４２】図14は、それぞれ16バイトのラインを有し、４KBインストラクション キャッシュ18をバックする４・エントリー インストラクション ストリーム バッファ62と、４KB データ キャッシュ20をバックするデータ ストリーム バッファ62のパフォーマンスを示す。この図は、バッファが最初のミスで開始して、プレフェッチを許されるライン数に基づき、除かれるミスの累計数を示すものである。ほとんどのインストラクションの参照（レファレンス）は、６番目の連続ラインがフェッチされる時までに、純粋な順番アクセス パターンを破り、一方多くのデータ参照パターンはより早く終結する。これの例外は、“liver ”に対するインストラクション参照及び“linpack ”に対するデータ参照である。“liver ”は、プログラムの14のループが順次実行され、初めの14ループは一般に他の手続きをコールしないか、あるいは過剰のブランチを行い、連続ミス パターンを破るため、変則である可能性がある。“linpack ”のデータ参照（レファレンス）パターンは次の如くして理解できる。ストリーム バッファ62は、キャッシュ18または20がミスしたラインを提供するのみの責務を有している。“linpack”の内側のループ（例えば、saxpy ）は、マトリックスの１行と他の行との間の内積（inner product ）を行う。１つの行の第１の使用により、この行はキャッシュにロードされる。キャッシュの当該の後続のミス（第１行のマッピング コンフリクトを除く）の後、マトリックスの次位のラインが構成される。マトリックスは過大であって、オン・チップ キャッシュに適さないので、全マトリックスは各反復（iteration ）毎にキャッシュを通過する。ストリーム バッファ62は第２レベル キャッシュ26によって提供される

20

最大帯域幅でこれを行いうる。基準ストリームがユニット・ストライドであるか、または各第３ワードまたは最大で他へスキップすることはこれに対し必須の要件である。非・ユニット・ストライド方向にアレイがアクセスされると、（さらに、他のディメンションが、ノン・トライビアルである限り）ここに説明したストリーム バッファ62は価かな利点しか有さなくなる。

【００４３】図15は、３つの典型的なストリーム バッファにおける帯域要求を示すものである。“ccom”に対するＩ・ストリームは極めて正規である。（インストラクション中に測定したとき。）平均として、各4.2 インストラクション毎に16B ラインをフェッチするを要する。プログラムが短いループに入ると、ストリーム バッファ62を参照する間隔が増加し、例えば疑問（else）クローズをスキップするような場合の如く、プログラムが小幅の前方ジャンプを行うようなときはこれは減少する。それにも拘らず、フェッチ周波数は極めて規則正しい。このデータは、例えばディジタルイクイップメント コーポレイション マルチチタン CPU またはMIPS R2000（商品名）の如く、短い機能ユニット レーテンシー（回転待ち時間）を有するマシンに対するものである。従ってインストラクション当りのサイクル数は、キャッシュ ミスの無いとき、１に極めて近くなる。

【００４４】“linpack ”及び“ccom”に対するデータ ストリーム バッファの参照のタイミングを図15に示してある。“linpack ”に対する新規な16B ラインの参照速度は、各27インストラクション当り、１である。“linpack ”のこの部分は二重精密（doubk precision ）であるため、この作業は、各13.5インストラクション当り、内側ループの新しい反復（iteration ）を行う。これは希望値よりも大である。この“linpack”のバージョンはある程度ルーズであり、各アレイ素子に対するアドレス計算を整数倍し、ループはアンロールされない。ループがアンロールされ、広範な最適化（optimization）が行われるとすると、参照のレート（比率）は増加する。しかし、インストラクション側には存しないデータ側の記憶トラフィックに起因するインストラクション ストリームのレートよりも以下でなければならない。“ccom”は興味あるトライモード（三モード）のパーフォーマンスを有している。ミスに続いて、後続のラインが使用されるとすると、平均でミス後、僅か５サイクルで必要とされる。ミス後次の２つのラインに対し、平均で各10インストラクション毎に連続するデータ ライン（16B ）が必要とされる。初めの３つのラインがストリーム バッファ62の殆どの（82％）利点を生ずる。その後は連続するラインは“linpack ”に近いレート、すなわち平均で各24インストラクション毎に必要とされる。

【００４５】一般に、バック・アップのメモリが新規ワード（４Ｂ）の各サイクルに平均帯域幅でデータを形成

することができると、ストリーム　バッファ62は連続参照を維持することが可能となる。これはインストラクション　ストリームに対し充分であり、かつ極めて多く巻戻し（アンロール）されたブロックコピーにも充分で、二重プレシジョン　ロード及びメモリ（store ）を使用する。この帯域幅が得られないときは、インストラクション　ストリームバッファの利点は減少し、ブロックコピー及び他の類似のオペレーションも負のインパクトを受ける。しかし各1.5 ～２サイクル当り１つの新しいワードを均等化（イコーリング）する帯域幅は多くのデータに対し依然として充分役に立つ。なおこれらの値は帯域幅に対するものであり、図12のプレフェッチ　スキムにおいて要求される全レーテンシィを達成するよりも遙に容易である。

【００４６】前章に述べたストリーム　バッファ62は、インストラクション　キャッシュ18のミスを全体で72％取除くことができた。しかしデータ　キャッシュ20のミスは、その25％しか取除くことができない。この理由の１つは、データの参照が、異なるデータ源よりのインタリーブされたデータ　ストリームで構成されていることによる。データの参照におけるストリーム　バッファ62のパフォーマンスの向上を図るため、マルチ・ウエイ　ストリーム　バッファ62のシステム80をシュミレートした。（図16）　このシステム80は４個のストリーム　バッファ62を並列にして構成されている。何れのストリーム　バッファ62をもヒットしないデータキャッシュ20にミスが生ずると、最低頻度（least recently）でヒットされたストリーム　バッファ62をクリヤし（すなわちLRU 置換）、ミスのアドレスにおいてフェッチを開始する。

【００４７】図17はマルチ・ウエイ　ストリーム　バッファ　システム80を我々のベンチマーク　セットで動作させたときのパフォーマンスを示す。予期したように、インストラクション　ストリームによるパフォーマンスは本質的には変らなかった。これはインストラクション　ストリームに対しては、より簡単な単一ストリームバッファ　システム60で充分間に合うことを示している。しかし、マルチ・ウエイ　ストリーム　バッファ　システム80は、データ側では画期的な性能向上改良を示し、これは６つのプログラムに対しミスの43％を除去することができ、単一ストリーム　バッファ　システム60の性能のほぼ２倍であった。“liver ”のマトリクス動作が最大の改良（減少を７％より60％に変えた）を示したが、すべてのプログラムが或程度の改良を示した。“liver ”もそのデータ構造に、ユニット　ストライド　アクセスを行うことを付記する。

【００４８】以上の説明においては、ストリーム　バッファ62に対し、単に１つのアドレスコンパレータのみが設けられていた。これは、要求されたラインがストリームバッファ62内であるが、コンパレータ72に対する第１

位置にない場合には、ストリーム　バッファ62は参照の際にミスをし、その内容はフラッシュされることを意味する。このスキムに対し、明らかな改良を行う１つは、コンパレータをストリーム　バッファ62の各位置に設けることである。かくすると、例えば単連続参照パターンによって、あるキャッシュ　ラインがスキップされても、ストリーム　バッファ62は、キャッシュ　ラインが既にフェッチされている限り依然としてこれに供給を行うことが可能である。

【００４９】図18は、３つの比較器（comparators）を持つストリーム　バッファの性能を示す。疑似ストリーム　バッファ（quasi-stream buffer）は、命令キャッシュ　ミスの76％を除去することができ、これは純粋逐次（purely sequential）ストリーム　バッファよりも４％の改善であって、残留ミスの数の14％の減少をもたらす。このことは恐らく、"if"ステートメント中の"then"クローズや"else"クローズのようなコードがスキップしたときに、疑似ストリーム　バッファが有用なフェッチを継続できる能力に依るものであろう。シミュレートされたこの変形は３つの比較器を持つので、最大２つのキャッシュ　ラインに加えて更に 3/4までのキャッシュ　ラインを、アラインメントによりどちらかの側で、最大合計16ないし22の命令に対してスキップすることが出来た。このことは、ストリーム　バッファがフラッシュされることの起きない（分枝アラインメントによる）逐次ストリーム　バッファでは、僅かに０ないし６命令しかスキップしないのに対比される。

【００５０】疑似ストリーム　バッファの余分の比較器はまた、４方向（four-way）データストリーム　バッファの性能をも改善する。結局全体では、４方向疑似ストリーム　バッファはすべてのミスの47％を除去することができ、これは純粋逐次４方向ストリーム　バッファよりも４％の向上である。

【００５１】単一ストリーム　バッファへ数個の余分な比較器を設けるために所要のハードウェアの量は小さいものだから、疑似ストリーム　バッファは、命令ストリームに対する逐次ストリーム　バッファの有益な一般化であるかのように思われる。それは、僅かに２つの比較器を付加することが、逐次ストリーム　バッファを疑似ストリーム　バッファに転換するのに要求されるだけだからである。しかし、多方向（multi-way）データ疑似ストリーム　バッファに対しては、それは有用ではないかも知れない、と云うのは所要の余分な比較器の数が何倍にも大きくなるであろうからである。ソフトウェア探究での興味ある分野として、コンパイラがコードを再編成し、データ　レイアウトがストリーム　バッファの用途を最大にする能力がある。もし引照規準の逐次性（sequentiality of references）を最適化する技術が成功するならば、ストリーム　バッファへの余分な比較器の必要性はさらに小さくなる。

23

【００５２】ストリーム　バッファの性能を視野に収めるために、このセクションではストリーム　バッファの性能を以前に文献で検討したプリフェッチ技術に置えることにする。我々の６つのベンチマーク上での、ミスの際のプリフェッチ(prefetchon miss)、タグ付プリフェッチ(tagged prefetch)　及び常時プリフェッチ(always prefetch)　の性能が表３に示される。このデータは、１命令付与(oneinstruction-issue)　の第２レベルキャッシュ回転待ち時間(latency)　を伴うこれらのプリフェッチ技術の使用を前提にして、ミスの減少を示している。１命令付与の回転待ち時間は１マシーンサイクルより小さいであろうから、また第２レベルキャッシュは典型的に多くの CPUサイクルの回転待ち時間を持つのだから、このデータは全く非現実的なものであることに留意されたい。それにも拘らず、これらの数字はこれらのプリフェッチ技術の性能の上限を与えるのである。この検討におけるプリフェッチ　アルゴリズムの性能は以前の文献に示されたデータとよく一致している。上に引用したスミス(Smith)　の論文では、16Bラインと８方向セット結合性(16B lines and 8-way set ass ociativity)　を伴う 8KB混合キャッシュ上のPDP-11痕跡TRACE(a PDP-11 tra ce TRACE on a 8K Bmixed cache)に対するミス　レート(miss rate)　の減少は（混合キャッシュだけしか検討されていないので）ミスの際のプリフェッチに対して27.8％、タグ付プリフェッチに対して50.2％、常時プリフェッチに対して51.8％であることが判っている。

24

＊【００５３】表４では、表４からのプリフェッチの性能を以前に示したストリーム　バッファの性能と比較している。命令の側では、単純単一ストリーム　バッファ(a simple single s tream buffer)62はミスの際のプリフェッチを広いマージンで出力実行している。このことは、純粋逐次引照規準ストリーム(a purely sequen tial reference stream)に対してミスの際のプリフェッチはミスの数を因数２で減少させるのみであろうから、驚くには当たらない。単純単一ストリーム　バッファシステム60も疑似ストリーム　バッファ　システム80も共に、タグ付プリフェッチと殆ど同じように機能している。トラフィックに関する限り、ストリームバッファ62はミスの後にタグ付プリフェッチより多くフェッチするであろうが、しかしタグ転移に際してはフェッチを開始しないだろうから、トラフィック率の比較は今後の興味ある研究課題である。命令ストリーム上のストリーム　バッファ62の性能は常時プリフェッチより僅かに劣る。このことは、常時プリフェッチの性能が分枝を取らない命令の百分率に近似するから、また命令の減少に際し逐次プリフェッチによるキャッシュ　ミスの上限だから、驚くには当たらない。しかし、ストリーム　バッファ62によるアプローチのトラフィック率は、常時プリフェッチよりもミスの際のプリフェッチ又はタグ付プリフェッチのそれに遙かに近いに相違ない。

【００５４】

【表３】

fetch	ccom	yacc	met	grr	liver	linpack	
4KB instr. cache, direct-mapped, 16B lines, 1-li							
prefetch latency:							
on miss	44.1	42.4	45.2	55.8	47.3	42.8	4
tagged	78.6	74.3	65.7	76.1	89.0	77.2	7
always	82.0	80.3	62.5	81.8	89.5	84.4	8
4KB data cache, direct-mapped, 16B lines, 1-instr prefe							
latency:							
on miss	38.2	10.7	14.1	14.5	45.8	75.7	3
tagged	39.7	18.0	21.0	14.8	63.1	83.1	4
always	39.3	37.2	18.6	11.7	63.1	83.8	4

【００５５】

【表４】

technique	misses eliminat
for 4KB direct-mapped instruction cache w/16B lines:	
Prefetch on miss (1-instr latency)	46.3
single stream buffer	72.0
quasi-stream buffer (3 comparator)	76.0
tagged prefetch (1-instr latency)	76.8
always prefetch (1-instr latency)	80.1
for 4KB direct-mapped data cache w/16B lines:	
single stream buffer	25.1
prefetch on miss (1-instr latency)	33.1
tagged prefetch (1-instr latency)	40.1
always prefetch (1-instr latency)	42.1
4-way stream buffer	43.1
4-way quasi-stream buffer	47.1

25

【００５６】表４はまた、ストリーム　バッファ62の性能を、データ引照のためのプリフェッチ技術に譬える。茲ではすべての型式の４方向ストリーム　バッファ　システム80がプリフェッチ戦術を出力実行する。これは主として、プリフェッチ戦術が、たとえそれが必要でない場合にさえも、プリフェッチされた項目を常にキャッシュ内に置くからである。ストリーム　バッファ62によるアプローチは、要請されたときにのみ項目をキャッシュ20内に動かすので、結果としてプリフェッチされたデータを常にキャッシュ20内に置くよりも汚染が少ない。このことは、データ引照規準の空間的所在が命令引照規準の空間的所在よりも少ないのだから、またプリフェッチされたデータはプリフェッチされた命令よりも遙かに汚染となり易いのだから、データ引照にとって特に重要である。

【００５７】ストリーム　バッファ62の相対的性能及び理想的なプリフェッチ技術とは無関係に、ストリーム　バッファによるアプローチは遙かに実行し易いものである。それはこのアプローチが（逐次引照規準パターンに対するミスの際のプリフェッチ又はタグ付プリフェッチとは異なり）パイプライン化されたメモリ　システムの利点を活用できるからである。それはまた、先行のブロックが使用される前にブロックをフェッチすることを開始できるから、プリフェッチされたデータに対する回転待ち時間への要求条件がプリフェッチ技術に較べて低い。最後に、少なくとも命令のストリーム　バッファ62に対しては、ストリーム　バッファ62が特別に必要とする余分のハードウェアは、タグ付プリフェッチが必要とする余分なタグ記憶部としばしば同程度である。

【００５８】小さいミス　キャッシュ42（例えば 2～8 エントリーの）は、1Kないし8Kバイトの範囲の直接写像されたキャッシュ20に対するデータ　キャッシュ衝突ミスの減少に効果的であることが示されている。それらは、キャッシュ中の同じラインに写像する２ないし４ライン間にミスが交互に生じる厳しい衝突を、効果的に除去する。

【００５９】ビクティム　キャッシュ52は、小さい付属キャッシュ52の内のキャッシュ　ミスのビクティムを、ターゲットに代わって救済するミスキャッシュ化の改良である。ビクティム　キャッシュ52は、衝突ミスを除去するのにミス　キャッシュ42よりもずっと効果的である。ビクティム　キャッシュ52は、ラインのサイズが増大し、衝突ミスの百分率が増加するのに伴い一層有益である。衝突ミスの百分率が増加するのに伴い、ビクティム　キャッシュ52により除去可能なこれらのミスの百分比も増加し、その結果は、ビクティム　キャッシュ52の使用により可能な性能の改善に対し更に険しい傾斜となるように、一般的には見受けられる。

【００６０】ストリーム　バッファ62は、ミスしたキャッシュ　ラインの後のキャッシュラインをプリフェッチ

26

する。ストリーム　バッファ62は該ラインを、不必要なキャッシュの汚染を避けるために（キャッシュ　ミスがもしあれば）キャッシュミスにより要請されるまで記憶しておく。それは容量及び強制的なミスの数を減少するのに特に有用である。それは、前に論じたタグ付プリフェッチ又はミスの際のプリフェッチのようなプリフェッチ技術とは異なり、逐次引照規準に対するパイプライン化されたメモリ　システムで使用可能なメモリ帯域幅を活用できる。ストリーム　バッファ62はまた、他のプリフェッチ技術（常時プリフェッチさえも）より遙かに前以てデータをプリフェッチするから、さらに長いメモリシステム回転待ち時間を許容できる。ストリーム　バッファ62はまた、命令衝突ミスも同じく相対的に逐次性を持つ傾向がある故、命令衝突ミスを補償することもできる。

【００６１】多方向ストリーム　バッファ　システム80は、同時に発生するいくつかのストリームをプリフェッチできるところの一組のストリーム　バッファ62である。この検討では、プリフェッチ開始アドレスが LRU (least recently used －最長時間未使用）順ですべてのストリーム　バッファ62に亙り置き換えられる。多重経路ストリーム　バッファ62は、アレイ操作(array operations)におけるようないくつかの異なる巨大データ構造へのインターリーブされたアクセス(interleavedaccesses)を含むデータ引照に対し有用である。しかし、プリフェッチすることは逐次ライン(sequential lines)だから、（２又は３の）単位幅(unit stride)のアクセス　パターンのみが利益を受ける。

【００６２】ビクティム　キャッシュ52による性能の改善及びストリームバッファ62による性能の改善は、データ引照規準に対し相対的に直交(relatively orthogonal)するものである。ビクティムキャッシュ52は、引照規準がキャッシュ中の同じラインに写像する２つの所在位置を交互に往き来するときに好適に働く。それはデータをプリフェッチする訳ではなく、フェッチされたデータを使用可能に維持するためにずっとよい仕事をするだけである。しかるに、ストリーム　バッファ62は、データをプリフェッチすることにより性能の改善を達成する。それは衝突が時間的に広い間隔を持つのでない限り衝突ミスを除去しない、そしてキャッシュ　ミス引照ストリームは多数の逐次アクセスから成るのである。これらは正に、その相対的に小さい容量の故にビクティム　キャッシュ52により旨く処理されない衝突ミスなのである。６つのベンチマークの一組に亙って平均的に、４エントリーのビクティム　キャッシュに当たった 4KBの直接写像されたデータ　キャッシュ　ミスの僅かに 2.5％が、ccom, met, yacc, grr 及び livermore に対して、４方向ストリーム　バッファにも当たるのである。対照的に、linpack はそのデータ　アクセス　パターンの故に、ビクティム　キャッシュに当たったものの50％が４方向スト

27

リーム　バッファにも当たるのである。しかし、linpack のキャッシュ　ミスの僅かに４％がビクティム　キャッシュに当たる。それは６つのベンチマークのうちでビクティム　キャッシュすることからもたらす利益が最小であり、従ってこのことはまだストリーム　バッファとビクティム　キャッシュすることとの間の有意な量の重複ではない。

【００６３】図20は、ベース　システム10に４エントリー　データビクティム　キャッシュ52と命令ストリーム　バッファ62と４方向ストリーム　バッファ62部分システムとを付加したものの性能を示すグラフで、該システムは図19A 及び図19B にシステム100 とされているものである。（該ベース　システムは、24サイクル　ミス　ペナルティズを持つオンチップ 4KB命令及び 4KBデータ　キャッシュ、並びに 128バイト　ライン及び320 サイクル　ミス　ペナルティを持つ３段階第２レベル１MBキャッシュへの16バイト　ラインを持つ。）図20の下の実線はビクティム　キャッシュ又はバッファを持たない原のベース　システムの性能を表し、上の実線はバッファ及びビクティム　キャッシュを持つ場合の性能を表している。これらの技術の組合せは第１レベルのミス　レートを、これらの特徴を持たないベースライン　システム10のそれの半分未満に減少させ、結果として６つのベンチマークに亘る平均で 143％のシステム性能の改善がもたらされている。これらの結果は、僅かの量のハードウエアの付加によりキャッシュ　ミス　レートを劇的に減少させ、システム性能を改善したことを示している。

【図面の簡単な説明】

【図１】図１は、本発明に係わるようなベースライン　システムの概略ブロック図である。

【図２】図２は、本発明に係わらない図１のシステムの性能を示すグラフである。

【図３】図３は、本発明に係わらない図１のシステムの性能の別の面を示すグラフである。

【図４】図４は、本発明の実施例に係わる図１に示すシステムの一部分の概略ブロック図である。

【図５】図５は、図４の部分に含まれるシステムの性能のある１つの面を示すグラフである。

【図６】図６も、図４の部分に含まれるシステムの性能のまた別の面を示すグラフである。

【図７】図７は、本発明のもう１つの実施例に係わる図４の部分に対応するシステムの一部分の概略ブロック図である。

【図８】図８は、図７の部分に含まれるシステムの性能のある１つの面を示すグラフである。

【図９】図９も、図７の部分に含まれるシステムの性能のある１つの面を示すグラフである。

【図１０】図10も、図７の部分に含まれるシステムの性能のある１つの面を示すグラフである。

【図１１】図11も、図７の部分に含まれるシステムの性

28

能のある１つの面を示すグラフである。

【図１２】図12は、本発明に係わらない図１のシステムの性能のもう１つの別の面を示すグラフである。

【図１３】図13は、本発明の更にもう１つの実施例に係わる図４の部分に対応するシステムの一部分の概略ブロック図である。

【図１４】図14は、図13の部分に含まれるシステムの性能のある１つの面を示すグラフである。

【図１５】図15も、図13の部分に含まれるシステムの性能のまた別の面を示すグラフである。

【図１６】図16は、本発明の更にもう１つの実施例に係わる図13の部分に対応するシステムの一部分の概略ブロック図である。

【図１７】図17は、図16の部分に含まれるシステムの性能のある１つの面を示すグラフである。

【図１８】図18も、図16の部分に含まれるシステムの性能のまた別の面を示すグラフである。

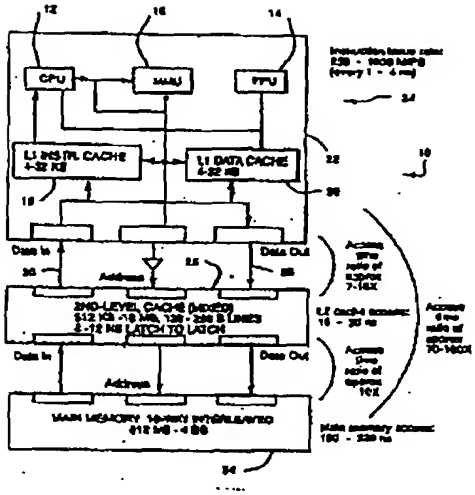
【図１９】図19は、図19A と図19B とを左右に並べたものであり、図19A と図19B とはそれぞれ、本発明の更にもう１つの実施例を示す図７、図13及び図16のシステム部分の組合せに一般的に対応するシステムの一部分の概略ブロック図の左半分と右半分とを示す図である。

【図２０】図20は、図19A と図19B のシステムの性能のある１つの面を示すグラフである。

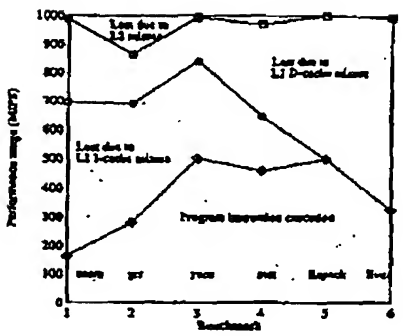
【符号の説明】

10　メモリ　システム（ベースライン　システム）
12　CPU （中央処理ユニット）
14　フローティング　ポイント　ユニット（FPU）
16　メモリ　マネージメント　ユニット（MMU）
18,20　データ　キャッシュ（第１レベル）
22　チップ（プロセッサ　チップ）
24　セントラル　プロセッサ（中央処理装置）
26　第２レベル　キャッシュ
28　ラッチ
34　主メモリ
40　ミス　キャッシュ　システム
42　ミス　キャッシュ
44　キャッシュ　ライン
50　キャッシュ　システム
52　完全付属ミス　キャッシュ（ビクティム　キャッシュ）
56　ライン
60　単一ストリーム　バッファ　システム
62　ストリーム　バッファ
64　エントリー
66　タグ
70　データ　ライン
72　比較器
80　多方向ストリーム　バッファ　システム

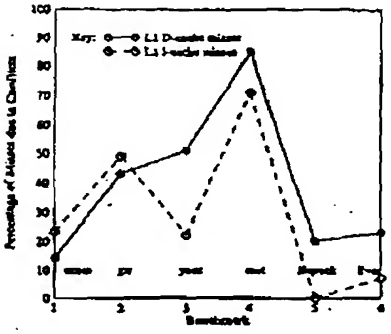
【図１】



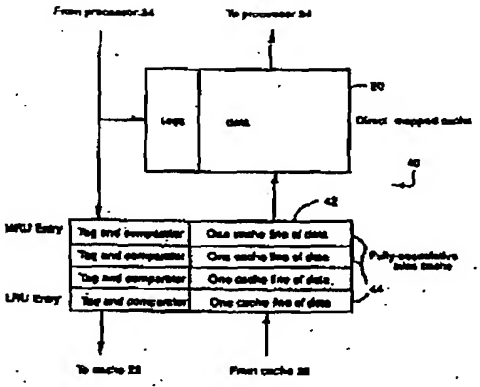
【図２】



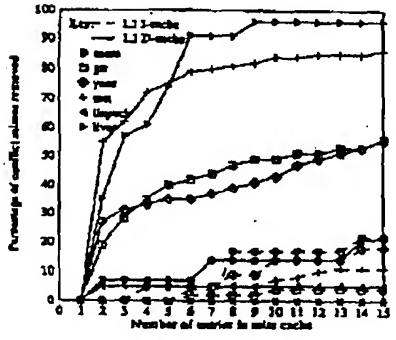
【図３】



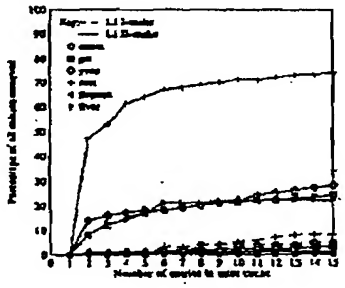
【図４】



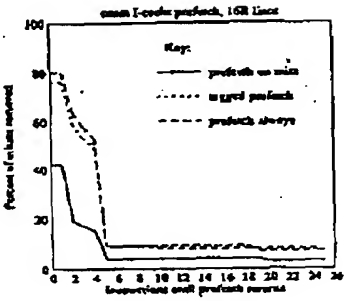
【図５】



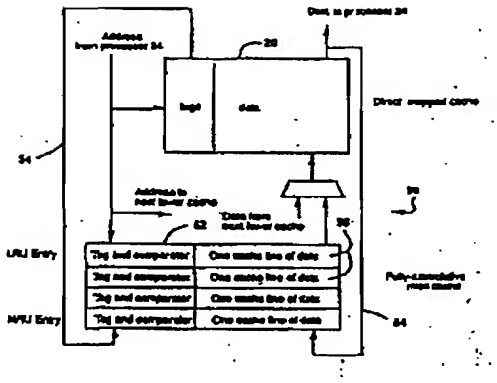
【図６】



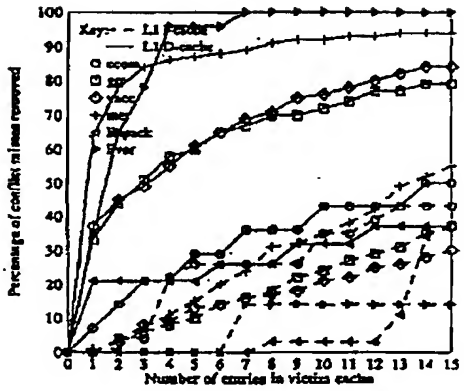
【図１２】



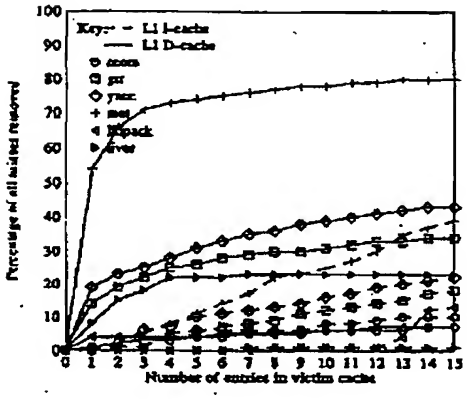
【図7】



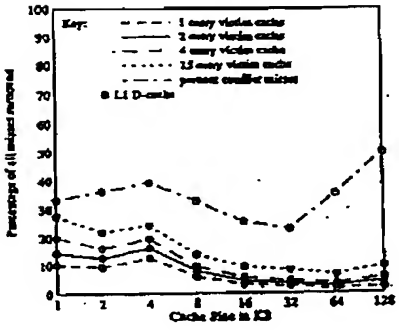
【図8】



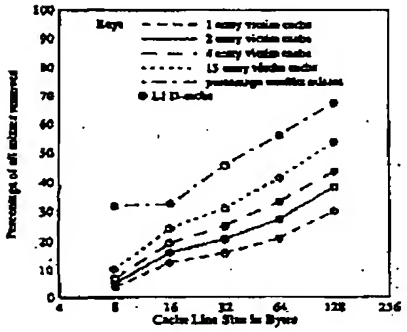
【図9】



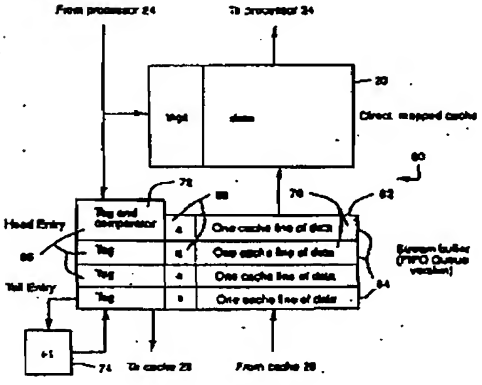
【図10】



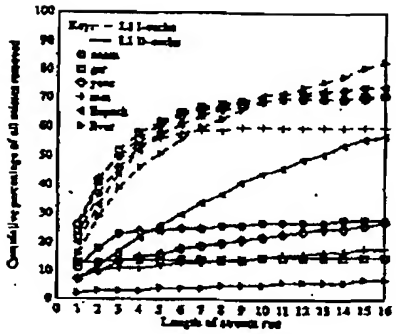
【図11】



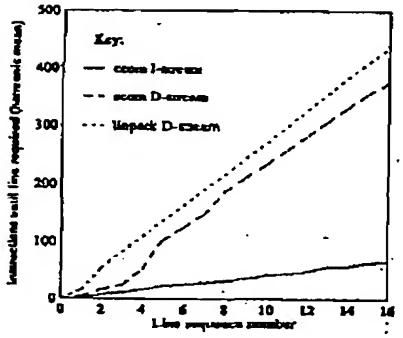
【図13】



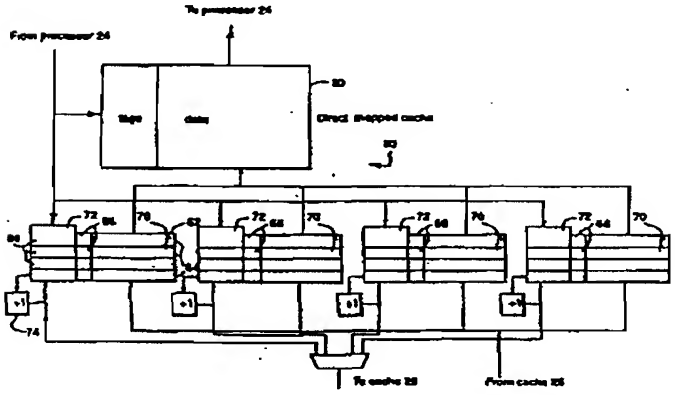
【図１４】



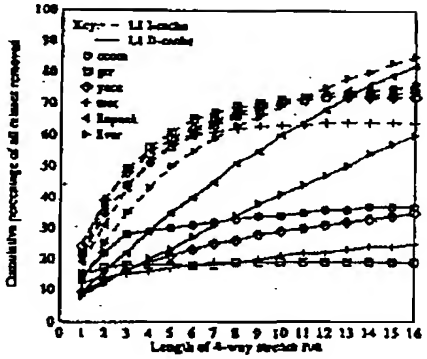
【図１５】



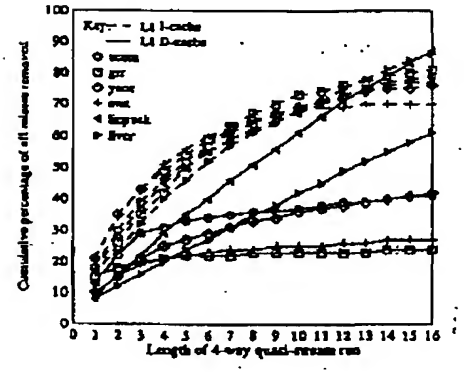
【図１６】



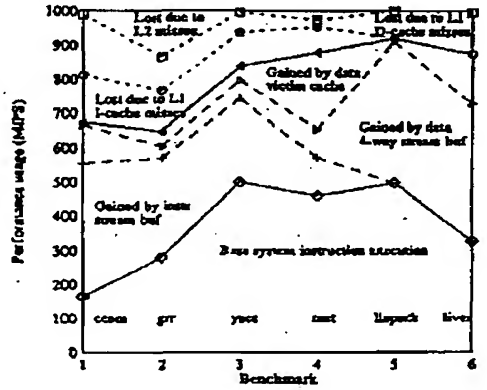
【図１７】



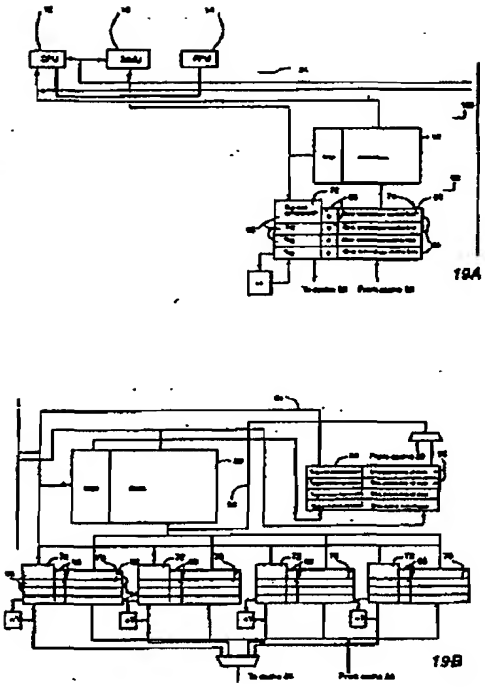
【図１８】



【図２０】



【図１９】



フロントページの続き

(72)発明者　アラン　ユーステース
アメリカ合衆国　カリフオルニア州
94302　パロ　アルト　ピー・オー・ボツ
クス　1605